

## PLP - Segundo Parcial - 2<sup>do</sup> cuatrimestre de 2017

Este examen se aprueba obteniendo al menos dos ejercicios bien (B) y dos regular (R), o con tres ejercicios bien menos (B-). Las notas para cada ejercicio son: M, R, B-, B. Se puede utilizar todo lo definido en las prácticas y todo lo que se dio en clase, colocando referencias claras.

### Ejercicio 1 - Programación lógica

Implementar los predicados respetando en cada caso la instanciación pedida. Los generadores deben cubrir todas las instancias válidas de aquello que generan sin repetir dos veces la misma. No usar cut (!) ni predicados de alto orden como `setof`, con la única excepción de `not`.

Durante este ejercicio, programaremos un simulador para la compañía aérea TODOSABORDO. Representaremos un avión con el functor `avion(N, Letras, Filas)` donde:

- `N` es un entero que representa la cantidad de filas de asientos que tiene el avión.
- `Letras` es una lista de átomos que determina los nombres de los asientos en cada fila.
- `Filas` es una lista de listas de tuplas (`D`, `P`) de tamaño `N`, donde `D` es un elemento de `Letras` y `P` es un átomo que representa el apellido del pasajero. Esta lista muestra la ubicación de cada pasajero en una fila. Los nombres de los asientos `D` respetan el orden en que figuran en la lista `Letras`.

Por ejemplo: `avion(2, [a, b], [[(a, perez), (b, garcia)], [(a, lopez), (b, fernandez)]])` representa un avión con 2 filas disponibles, con letras de asientos `a` y `b`, donde `perez` y `garcia` ocupan la primer fila; `lopez` y `fernandez` la segunda. En caso de que un asiento no esté ocupado, dicha letra no figura en la lista de `Filas`. Por ejemplo: `avion(2, [a, b], [[], []])` representa a un avión totalmente desocupado.

- a) Definir el predicado `filaMasVacía(+A, ?I)` que es verdadero cuando dado un avión `A`, `I`<sup>1</sup> es el número de fila que tiene más asientos desocupados. Si `I` está fuera de rango, el predicado debe dar `false`.

**Ejemplo:** `?- filaMasVacía(avion(2, [a, b], [[], []]), I)`

`I = 1; I = 2; false;`

- b) Definir el predicado `elegirPasajero(+N, +Ps, ?Es)` que dado un entero `N` y una lista sin repetidos de átomos `Ps`, es verdadero cuando `Es` es una posible manera de elegir `N` pasajeros de `Ps` en algún orden.
- Ejemplo:** `?- elegirPasajero(2, [perez, garcia, lopez], Es)`

<code>Es = [perez, garcia];</code>	<code>Es = [perez, lopez];</code>	<code>Es = [garcia, lopez];</code>	<code>false;</code>
<code>Es = [garcia, perez];</code>	<code>Es = [lopez, perez];</code>	<code>Es = [lopez, garcia];</code>	

- c) Definir el predicado `ocupar(+N, +Ls, +Ps, ?A)` que dado un entero `N` y dos listas sin repetidos de átomos `Ls` y `Ps`, es verdadero cuando `A` es un avión de `N` filas con `Ls` letras, ocupados por pasajeros pertenecientes a `Ps`. Notar que la lista de pasajeros podría ser menor o mayor a la cantidad de asientos disponibles. Además, no es necesario que todos los asientos estén ocupados, ni que todos los pasajeros se suban al avión.

**Ejemplo:** `ocupar(2, [a, b], [perez, garcia, lopez], A)`

<code>A = avion(2, [a, b], [[], []]);</code>	<code>% avion vacio</code>
<code>A = avion(2, [a, b], [(a, perez)], []);</code>	<code>% perez en fila 1 asiento a</code>
<code>A = avion(2, [a, b], [(b, garcia)], []);</code>	<code>% garcia en fila 1 asiento b</code>
<code>...</code>	
<code>A = avion(2, [a, b], [[], [(b, perez)]]);</code>	<code>% perez en fila 2 asiento b</code>
<code>A = avion(2, [a, b], [(a, perez), (b, garcia)], [(b, lopez)]);</code>	<code>% fila 1 llena, lopez en fila 2 asiento b</code>
<code>...</code>	

<sup>1</sup>Comienza con el valor 1.

## Ejercicio 2 - Resolución

Dadas las siguientes fórmulas en lógica de primer orden del Cálculo Lambda con Subtipado:

- $\text{bool} <: \text{nat}$
- $\forall X, Y, Z \ X <: Y \wedge Y <: Z \supset X <: Z$  (S-Tran)
- $\text{nat} <: \text{int}$
- $\forall X, Y, Z, W \ Z <: X \wedge Y <: W \supset X \rightarrow Y <: Z \rightarrow W$  (S-Arrow)
- $\forall X \ X <: X$  (S-Ref)

- a. Convertir las fórmulas anteriores a forma clausal y, utilizando resolución, probar que la siguiente relación es verdadera:  $\text{int} \rightarrow \text{nat} <: \text{bool} \rightarrow \text{nat}$ . Indicar la sustitución utilizada en cada paso.
- b. Indicar si utilizó resolución SLD y/o resolución binaria.

## Ejercicio 3 - Programación objetos

Se llama *profiling* al hecho de analizar dinámicamente un programa para obtener métricas como: cantidad de memoria utilizada, tiempo de ejecución, etc. Entonces, un *Profiler* es un programa especial que se encarga de, dado un programa objetivo, realizar las mediciones pertinentes sobre éste.

```
Profiler subclass: #TimeProfiler
class > newForTarget: anObject
| startTime |
startTime := 0.
^self newForTarget: anObject
withSetup: [startTime := Time now]
withPos: [Time now subtractTime: startTime]:
```

```
escenario1
| timeProfiler totalTime |
timeProfiler := TimeProfiler newForTarget: 20.
totalTime := timeProfiler factorial.

escenario2
| timeProfiler maxTime |
timeProfiler := TimeProfiler newForTarget: 20.
maxTime := timeProfiler profile:
    #(Message selector: #factorial.
      Message selector: #isPrime.
      Message selector: #nthRoot argument: 10)
    accordingTo: [:a :b | a max: b].
```

- a) Implementar la clase **Profiler** que permita analizar de forma *genérica* distintas métricas para un objeto dado. Su implementación debe pasar el **escenario1**.
- b) Extender la clase **TimeProfiler** con el mensaje de instancia **profile:accordingTo:** que devuelve el tiempo del mensaje según el criterio definido en el bloque que se pasa como argumento. Su implementación debe pasar el **escenario2**.

## Ejercicio 4 - Subtipado

Se extiende el Cálculo Lambda para soportar **matrices estáticas**, creadas con un tamaño fijo y sabiendo que no pueden ser redimensionadas.

$M ::= \dots \mid \text{new}(\underline{n}, \underline{m}, M) \mid M[\underline{n}][\underline{m}] \leftarrow N \mid M[\underline{n}][\underline{m}]$  donde  $\underline{n}$  abrevia  $\text{succ}^n(0)$

$\sigma ::= \dots \mid \sigma_{n \times m}^*$

$$\frac{\Gamma \triangleright M : \sigma \quad \Gamma \triangleright \underline{n} : \text{Nat} \quad \Gamma \triangleright \underline{m} : \text{Nat}}{\Gamma \triangleright \text{new}(\underline{n}, \underline{m}, M) : \sigma_{n \times m}^*} \text{(T-NEW)} \quad \frac{\Gamma \triangleright M : \sigma_{n \times m}^* \quad \Gamma \triangleright \underline{i} : \text{Nat} \quad \Gamma \triangleright \underline{j} : \text{Nat}}{\Gamma \triangleright M[\underline{i}][\underline{j}] : \sigma} \text{(T-GET)}$$

$$\frac{\Gamma \triangleright M : \sigma_{n \times m}^* \quad \Gamma \triangleright \underline{i} : \text{Nat} \quad \Gamma \triangleright \underline{j} : \text{Nat} \quad \Gamma \triangleright N : \sigma}{\Gamma \triangleright M[\underline{i}][\underline{j}] \leftarrow N : \sigma_{n \times m}^*} \text{(T-ASIGN)}$$

Notar que  $\sigma_{n \times m}^*$  indica que habrá un tipo distinto para cada matriz de tipo  $\sigma$  y tamaño  $n \times m$ . En el momento de la definición *new*, además de indicarse las filas y columnas de la matriz, se debe indicar el elemento *por defecto*  $M$ , que será el contenido de las posiciones aún no asignadas.

- a. Extender la definición de subtipado para el tipo de las matrices estáticas, justificando sus decisiones en función del principio de substitutividad, de manera tal que la siguiente expresión sea bien tipada:

$$\emptyset \triangleright (\lambda x : \text{Nat}_{2 \times 2}^*. x[1][1] \leftarrow x[2][2]) \text{new}(\underline{3}, \underline{3}, \text{false}) : \text{Nat}_{2 \times 2}^*$$

- b. Exhibir una derivación para el juicio de tipado del inciso anterior.



EDGARDO				
7	2	3	4	N
B	B	-	B	A

Sebastián Taboh  
30

1/5

21/11/17.

①

a) filaMásVacía (+A, ?I)

ESTO SE PUEDE  
NO LEER. OK! :)

filaMásVacía (avión(1, -, -), 1).

filaMásVacía (avión(N, L, [FIFs]), 1) :-

N > 1, ~~CantOcupados~~ is CantOcupados (avión(N, L, [FIFs]),  
F),  
CantOcupados),

~~IndiceFilaMásVacía is~~

NuevaCantFilas is N-1,

filaMásVacía (avión(nuevaCantFilas, L, Fs),  
IndiceFilaMásVacía),

~~FilaAComparar~~

nth1 (IndiceFilaMásVacía, Fs, FilaAComparar),

cantOcupados(~~avión~~ FilaAComparar, OtraCantidad)

cantOcupados =< OtraCantidad.

donde  $\uparrow$  cantOcupados es el predicado length.  
el predicado

Esto así como está, está incompleto.

Voy a hacer otra cosa. Voy a hacer un "map" para  
obtener las cantidades de ocupados por fila, voy a  
obtener el mínimo, obtener la lista de índices que  
cumplan que en esa posición el valor es = al mínimo,  
y hacer member.

ocupadosPorFila ([F], [X]) :- length(F, X).

ocupadosPorFila ([F|Fs], [X|Xs]) :-

length(F, X), ocupadosPorFila (Fs, Xs). ✓

Uso minLista del Ejercicio 5 de la Práctica 5. ✓

LEER

obtenerIndices( $L, -, [ ]$ ).  
obtenerIndices( $[H|Ms], M, Is$ ):-

obtenerIndices( $Ms, M, Is$ ):-

obtenerIndices( $[H|Ms], M, Is, 1$ )

obtenerIndices( $[H], M, ~~[ ]~~ [K], K$ )

obtenerIndices( $[H|Ms], M, Is, K$ ):-

% appendes k a obtenerIndices( $Ms, M, K+1$ ).  
NK is  $[K]$ , obtenerIndices( $Ms, M, L_2, NK$ ),  
append( $L_1, L_2, Is$ ).

filaMasVacía( $aviso(1, -, -, 1)$ ).

filaMasVacía( $aviso(N, L, ~~[ ]~~ Fs), I$ ):-

ocupadosPorFila( $Fs, Cs$ ), ✓

minLista( $Cs, M$ ), ✓

obtenerIndices( $Cs, M, Is, 1$ ), ✓

member( $I, Is$ ). ✓

obtenerIndices( $+Cs, +M, -Is, +K$ )

obtenerIndices( $[H], M, [K], K$ )  $\rightarrow$  obtenerIndices( $[NoM], M, [ ], -$ ):-  $\begin{matrix} NoM \\ K \\ M \end{matrix}$

obtenerIndices( $[H|Ms], M, Is, K$ ):-

$L_1$  is  $[K]$ , NuevoK is  $K+1$ ,

obtenerIndices( $Ms, M, L_2, NuevoK$ ), ✓

append( $L_1, L_2, Is$ ). ✓

obtenerIndices( $[NoM|Ms], M, Is, K$ ):-

$NoM = 1 = M$ , NuevoK is  $K+1$ ,

obtenerIndices( $Ms, M, ~~L_2~~, NuevoK$ ), ✓

~~append( $[ ], L_2, Is$ ).~~  $Is$



- b) Como la lista no tiene repetidos, puede pensarse que es un conjunto y que se piden los subconjuntos de cardinal  $N$ . **ENTODOS LOS ORDENES POSIBLES.**

Esto es igual al ejercicio 11 de la Práctica 5 excepto porque en ese se preservaba el orden, habría que dar las permutaciones de esas soluciones.

$\text{elegirPasajero}(+N, +Ps, ?Es)$

$\text{elegirPasajero}(N, Ps, Es) :-$

$\text{elementosTomadosEnOrden}(Ps, N, \text{AlgunaEs}),$

$\text{permutación}(Es, \text{AlgunaEs}, Es).$

$\text{permutación}(+L1, ?L2)$

•  $\text{permutación}([], [])$ .

$\text{permutación}([X|XS], [X|YS]) :- \text{permutación}(XS, YS)$

$\text{permutación}([X|XS], [NoX|YS]) :-$

$X \neq NoX,$

$\text{apariciones}(X, XS, Ap1), \text{ApIzg} \text{ is } Ap1 + 1,$

$\text{apariciones}(X, YS, ApIzg),$

$\text{apariciones}$

•  $\text{permutación}([X|XS], L) :-$

$\text{apariciones}(X, XS, \text{ApEnCola}),$

$\text{ApIzg} \text{ is } \text{ApEnCola} + 1,$

$\text{apariciones}(X, L, \text{ApIzg}),$

~~Resto~~  $\text{delete}(L, X, L2),$

$\text{delete}(XS, X, \text{RestoDeLaIzg}),$

$\text{permutación}(\text{RestoDeLaIzg}, L2).$

~~ESTO ESTÁ INCOMPLETO.~~

ESTO ES DIFÍCIL, VAMOS A HACERLO DE

OTRA FORMA.

Además, me faltaría copiar apariciones (la hice en la seria para el de las recetas).

Otra opción es generar todas las listas de números entre 1 y la longitud de  $P_s$  (generate),  
y en la etapa de test ver que no tengan repetidos  
y hacer una especie de map usando nth1.

elegirPasajero( $N, P_s, E_s$ ): -

length( $P_s, L$ ), generate( $N, L$ ), test(List),  
map(List,  $P_s, E_s$ ).

generate(0, -, []).

generate( $N, L, List$ ): -  $N > 0$ , length(List,  $N$ ),

NuevoN is  $N - 1$ , generate(NuevoN,  $L, List2$ ),

between(1,  $L, X$ ), List1 ~~is~~ [X],

append(List1, List2, List).

test(List): - sinRepetidos(List).

sinRepetidos([]).

sinRepetidos( $[X|Xs]$ ): - not(member( $X, Xs$ )),  
sinRepetidos( $Xs$ ).

map(+L, + $P_s$ , ? $E_s$ )

map([], -, []).

map( $[X|Xs], P_s, [P|Es]$ ): -

nth1( $X, P_s, P$ ), map( $Xs, P_s, Es$ ).

c) ocupar(+N, +Ls, +Ps, ?A)

A la suma va a haber Ps personas en el avión.

~~ocupar(N, Ls, Ps, avión(N, Ls, Fs)) :-  
ocuparFilas(Ps, Fs)~~

ocupar(1, Ls, Ps, avión(1, Ls, [F])) :-  
copio lo de abajo acá (lo recuadrado) ✓

~~ocuparFilas(Ps, Fs) :-~~

ocupar(N, Ls, Ps, avión(N, Ls, [F|Fs])) :- N > 1,

~~elegirPasajeros~~ length(Ls, CantAsientosPorFila),  
between(0, CantAsientosPorFila, CantGenteEnFila), ✓

~~ubicar~~

elegirPasajeros(CantGenteEnFila, Ps, G),  
ubicar(G, F, Ls),

borrar(~~G~~, ~~Ps~~, ~~GenteParaUbicar~~),  
Ps, G, ✓

NuevaN es N-1, ✓

ocupar(NuevaN, Ls, GenteParaUbicar,  
avión(NuevaN, Ls, Fs)). ✓

ubicar(+G, -F, +Ls)

ubicar([], [], -). ✓

ubicar(-, [], []). % Esto no hace falta, no va a pasar.

ubicar([G|Gs], [A|As],





borrar(+L1, +L2, -L3)

Dado que estas listas no tienen repetidos, el borrar que necesito es la resta de conjuntos que se sugiere hacer en el Ej-23 de la PS.

borrar(L, [ ], L).

borrar(L1, [X|XS], L) :- borrarElemento(L1, X, L2),  
borrar(L2, XS, L).

donde borrarElemento es el pedido en el Ej. 9, item III) de la PS.

⊗

ubicar tiene que generar todas las listas posibles de longitud igual a la longitud de G, donde estas duplas van a ser (letra de Ls, apellido de G), filtrar esas listas de modo que estén ordenadas por la primera componente

ubicar(G, A, Ls) :-

length(G, CantPersonas),

elementosTomadosEnOrden(Ls, CantPersonas, Letras),

obtenerAlguno(G, Persona), ~~obtenerAlguno(Letras, L)~~

Dupla is [(L, Persona)], nth1(1, Letras, L),

NO ESTÁ  
DEFINIDO

delete(G, Persona, NuevaG),

delete(Letras, L, NuevaLetras),

ubicar(NuevaG, A2, NuevaLetras),

append(Dupla, A2, A).



(2) a)

1)  $\{ \text{bool} <: \text{nat} \}$  ✓

2)  $\{ \text{nat} <: \text{int} \}$  ✓

3)  $\forall x. x <: x \leadsto \{ x_1 <: x_1 \}$  ✓

4)  $\forall x. \forall y. \forall z. (x <: y \wedge y <: z) \supset x <: z$

• Elimino la implicación:

$$\forall x. \forall y. \forall z. \neg(x <: y \wedge y <: z) \vee x <: z$$

• Paso a FNN:

$$\forall x. \forall y. \forall z. \neg x <: y \vee \neg y <: z \vee x <: z$$

• Paso a FNS: no hay nada que hacer.

• Ya se puede escribir en forma clausal

$$\{ \neg x_2 <: y_2, \neg y_2 <: z_2, x_2 <: z_2 \}$$

5)  $\forall x. \forall y. \forall z. \forall w.$

$$(z <: x \wedge y <: w) \supset x \rightarrow y <: z \rightarrow w$$

• Elimino la implicación:

$$\forall x. \forall y. \forall z. \forall w.$$

$$\neg(z <: x \wedge y <: w) \vee x \rightarrow y <: z \rightarrow w$$

• Paso a FNN:

$$\forall x. \forall y. \forall z. \forall w.$$

$$\neg z <: x \vee \neg y <: w \vee x \rightarrow y <: z \rightarrow w$$

• Ya no hay nada que hacer, se puede escribir en forma clausal.

$$\{ \neg z_3 <: x_3, \neg y_3 <: w_3, x_3 \rightarrow y_3 <: z_3 \rightarrow w_3 \}$$

Quiero probar que vale  $\text{int} \rightarrow \text{nat} <: \text{bool} \rightarrow \text{nat}$ .

Niego eso y lo refuto usando resolución.

6)  $\{ \neg \text{int} \rightarrow \text{nat} <: \text{bool} \rightarrow \text{nat} \}$  ✓

5 y 6 con  $\sigma_1 = \{ x_3 \leftarrow \text{int}, y_3 \leftarrow \text{nat}, z_3 \leftarrow \text{bool}, w_3 \leftarrow \text{nat} \}$

7)  $\{ \neg \text{bool} <: \text{int}, \neg \text{nat} <: \text{nat} \}$  ✓

7 y 3 con  $\sigma_2 = \{ x_1 \leftarrow \text{nat} \}$  ✓

8)  $\{ \neg \text{bool} <: \text{int} \}$  ✓

8 y 4 con  $\sigma_3 = \{x_2 \leftarrow \text{bool}, z_2 \leftarrow \text{int}\}$  ✓

9)  $\{ \neg \text{bool} <: y_2, \neg y_2 <: \text{int} \}$  ✓

9 y 1 con  $\sigma_4 = \{y_2 \leftarrow \text{nat}\}$  ✓

10)  $\{ \neg \text{nat} <: \text{int} \}$  ✓

10 y 2 con  $\sigma_5 = \emptyset$ . ✓

11)  $\square$ . ✓

b) Se utilizó resolución SLD dado que: ✓

- todas fueron cláusulas de Horn (a lo sumo un literal positivo). ✓
- se empezó por una cláusula objetivo (la 6)). ✓
- fue lineal. ✓
- se utilizó resolución binaria. ✓



4) a)

$$\frac{\tau <: \tau', n \leq n', m \leq m'}{\sigma_{n \times m}^* <: \tau_{n \times m}^*} \text{ (S-MAT)}$$

Intuición: si espero una matriz de tipo  $\tau$  de tamaño  $n$  por  $m$ , me viene bien una cuyo tipo sea subtipo de  $\tau$  (para poder realizar operaciones que necesite, por ejemplo) si además tiene una submatriz de tamaño  $n \times m$  (podría haber indexaciones que no andarían bien si esto no fuera así, por dar un ejemplo).

b)

Christian dijo que podía asumir que valía.

$$\text{(S-REFL)} \quad \frac{}{\text{Nat} <: \text{Nat}, 2 \leq 3, 2 \leq 3}$$

$$\text{(S-MAT)} \quad \frac{\text{Nat}_{3 \times 3}^* <: \text{Nat}_{2 \times 1}^*, \text{Nat}_{2 \times 2}^* <: \text{Nat}_{2 \times 2}^*}{\text{Nat}_{2 \times 2}^* \rightarrow \text{Nat}_{2 \times 2}^* <: \text{Nat}_{3 \times 3}^* \rightarrow \text{Nat}_{2 \times 2}^*} \text{ (S-REFL)}$$

$$\text{*4} \quad \frac{}{\text{Nat}_{2 \times 2}^* \rightarrow \text{Nat}_{2 \times 2}^* <: \text{Nat}_{3 \times 3}^* \rightarrow \text{Nat}_{2 \times 2}^*} \text{ (S-ARROW)}$$

\*3

\*4

\*1

$$\phi \triangleright \lambda x. \text{Nat}_{2 \times 2}^*. x[1][1] \leftarrow x[2][2]; \text{Nat}_{2 \times 2}^* \rightarrow \text{Nat}_{2 \times 2}^* \quad \text{Nat}_{2 \times 2}^* \rightarrow \text{Nat}_{2 \times 2}^* <: \text{Nat}_{3 \times 3}^* \rightarrow \text{Nat}_{2 \times 2}^* \text{ (T-SUB)}$$

$$\phi \triangleright \lambda x. \text{Nat}_{2 \times 2}^*. x[1][1] \leftarrow x[2][2] : \text{Nat}_{3 \times 3}^* \rightarrow \text{Nat}_{2 \times 2}^*$$

\*1

\*2

$$\phi \triangleright \lambda x. \text{Nat}_{2 \times 2}^*. x[1][1] \leftarrow x[2][2] : \text{Nat}_{3 \times 3}^* \rightarrow \text{Nat}_{2 \times 2}^*, \phi \triangleright \text{new} \langle 3, 3, \text{false} \rangle : \text{Nat}_{3 \times 3}^*$$

$$\text{(T-APP)} \quad \phi \triangleright (\lambda x. \text{Nat}_{2 \times 2}^*. x[1][1] \leftarrow x[2][2]) \text{new} \langle 3, 3, \text{false} \rangle : \text{Nat}_{2 \times 2}^*$$



$(\tau\text{-Bool}) \frac{}{\phi \triangleright \text{false} : \text{Bool}, \text{Bool} <: \text{Nat}}$   $(\tau\text{-Sub}) \frac{}{\phi \triangleright \text{false} : \text{Nat}, \phi \triangleright 3 : \text{Nat}, \phi \triangleright 3 < \text{Nat}}$   $(\tau\text{-New}) \frac{}{\phi \triangleright \text{new} \langle 3, 3, \text{false} \rangle : \text{Nat}^*_{3 \times 3}}$   
 Asumir que valen (o sea, son del 1º parcial).

Igual que ~~⊗~~  $(\tau\text{-Get}) \frac{}{\Gamma \triangleright x : \text{Nat}^*_{2 \times 2}, \Gamma \triangleright 2 : \text{Nat}, \Gamma \triangleright 2 : \text{Nat}}{\Gamma \triangleright x[2][2] : \text{Nat}}$   
 son del 1º parcial.

$(\tau\text{-Var}) \frac{}{\Gamma \triangleright x : \text{Nat}^*_{2 \times 2} \in \Gamma}$   $(\tau\text{-Assign}) \frac{}{\Gamma \triangleright x : \text{Nat}^*_{2 \times 2}, \Gamma \triangleright 1 : \text{Nat}, \Gamma \triangleright 1 : \text{Nat}, \Gamma \triangleright x[2][2] : \text{Nat}}{\Gamma \triangleright \{x : \text{Nat}^*_{2 \times 2} \} \triangleright x[1][1] \leftarrow x[2][2] : \text{Nat}^*_{2 \times 2}}$   $(\tau\text{-Ass}) \frac{}{\phi \triangleright \lambda x. \text{Nat}^*_{2 \times 2} \cdot x[1][1] \leftarrow x[2][2] : \text{Nat}^*_{2 \times 2} \rightarrow \text{Nat}^*_{2 \times 2}}$   
 Asumir que valen (son del 1º parcial).